Hindawi
Complexity
Volume 2021, Article ID 8192320, 12 pages
https://doi.org/10.1155/2021/8192320

Research Article

Study of Keyword Extraction Techniques for Electric Double-Layer Capacitor Domain Using Text Similarity Indexes: An Experimental Analysis

M. Saef Ullah Miah,[1] **Junaida Sulaiman**,[1,2] **Talha Bin Sarwar**,[3] **Kamal Z. Zamli**,[1] **and Rajan Jose**[4]

[1]*Faculty of Computing, College of Computing and Applied Sciences, Universiti Malaysia Pahang, Pekan 26600, Malaysia*
[2]*Center for Data Science and Artificial Intelligence (Data Science Center), Universiti Malaysia Pahang, Pekan 26600, Malaysia*
[3]*Department of Computer Science, Faculty of Science and Technology, American International University-Bangladesh (AIUB), Dhaka, Bangladesh*
[4]*Faculty of Industrial Sciences & Technology, Universiti Malaysia Pahang, Gambang 26300, Malaysia*

Correspondence should be addressed to Talha Bin Sarwar; talhasarwar40@gmail.com

Received 14 August 2021; Accepted 5 November 2021; Published 2 December 2021

Academic Editor: Lingzhong Guo

Keywords perform a significant role in selecting various topic-related documents quite easily. Topics or keywords assigned by humans or experts provide accurate information. However, this practice is quite expensive in terms of resources and time management. Hence, it is more satisfying to utilize automated keyword extraction techniques. Nevertheless, before beginning the automated process, it is necessary to check and confirm how similar expert-provided and algorithm-generated keywords are. This paper presents an experimental analysis of similarity scores of keywords generated by different supervised and unsupervised automated keyword extraction algorithms with expert-provided keywords from the electric double layer capacitor (EDLC) domain. The paper also analyses which texts provide better keywords such as positive sentences or all sentences of the document. From the unsupervised algorithms, YAKE, TopicRank, MultipartiteRank, and KPMiner are employed for keyword extraction. From the supervised algorithms, KEA and WINGNUS are employed for keyword extraction. To assess the similarity of the extracted keywords with expert-provided keywords, Jaccard, Cosine, and Cosine with word vector similarity indexes are employed in this study. The experiment shows that the MultipartiteRank keyword extraction technique measured with cosine with word vector similarity index produces the best result with 92% similarity with expert-provided keywords. This study can help the NLP researchers working with the EDLC domain or recommender systems to select more suitable keyword extraction and similarity index calculation techniques.

1. Introduction

Keywords are significant for automated document processing. Keywords are the concise representation of the contents of a document [1]. From keywords, the context of the documents can be easily understood. When there is a need to process lots of documents or classify any document for any purpose, it is tedious to go through the whole document one by one and classify them. Instead, going through the keywords makes this process faster, even for a human. However, it is also a time-consuming process to go through the keywords for many documents by a human. This task can be automated by employing machines to look for the keywords and classify the documents. Since the process of keyword extraction is being automated, it should also be assured that extracted keywords represent the actual context of the document; else automated extraction will be a complete loss of time and resources. This assurance can be done by comparing the extracted keywords with human or expert assigned keywords. Therefore, this paper introduces an experimental study to measure the similarity score between expert-provided keywords and keyword extraction algorithms generated keywords to observe how similar the machine-generated keywords' values are to the expert-provided keywords. In other words, this experiment can guide if the machine-generated keywords are feasible to utilize instead of expert-provided keywords for any specific domain.

There are several different keyword extraction algorithms available at present [2, 3]. These algorithms are employed in different scenarios, such as recommender systems, trend analysis, similar document identification, and relevant document selection [4–6]. All these algorithms are divided into three primary categories based on their extraction technique: supervised, unsupervised, and semi-supervised technique [7]. This study compares the similarity scores for supervised and unsupervised techniques with three prominent similarity indexes, namely, Jaccard similarity index [8], cosine similarity index [9, 10], and cosine with Word vector similarity [11]. The key contributions of this work are

(i) Recommending a keyword extraction technique that provides more similar machine-generated keywords to the expert or human provided keywords

(ii) Recommending type of texts (positive texts only or whole text of a document) that provides more similar keywords

(iii) Recommending a better similarity index for measuring similarity score between documents

(iv) Finding the feasibility of utilizing machine-generated keywords instead of expert-curated keywords

The rest of the paper is organized as follows. Employed keyword extraction techniques and relevant works are presented in Section 2 with their known shortcomings and strengths. Employed methodologies for the experiment are mentioned in Section 3. Then, the result analysis of the experiment is discussed in Section 4, and concluding remarks in Section 5.

2. Background Study

In this paper, some notable and well-known similarity index calculation algorithms and keyword extraction algorithms are employed. All the text similarity and keyword extraction algorithms with shortcomings and strengths are discussed in this section.

2.1. Keyword Extraction. Keyword extraction from text is an analysis technique that automatically extracts the most used and most important words or phrases from text based on different parameters [12]. In some techniques, these parameters can be defined externally, and some techniques do not support external definition [7]. Mainly there are three classes of keyword extraction techniques. Among them, supervised and unsupervised techniques are employed in this study.

2.1.1. Unsupervised Keyword Extraction. Four unsupervised keyword extraction techniques are employed in this paper. Unsupervised techniques are prone to poor accuracy and require a larger corpus input and do not extrapolate well [13]. However, unsupervised techniques are utilized widely compared to supervised techniques, as all sorts of domain-specific training labeled data are not always available for all the domains.

(1) YAKE. YAKE was proposed by Campos et al. [14]. It is a lightweight unsupervised keyword extraction technique based on TF-IDF. YAKE extracts keywords by calculating five features, namely, Word Casing (WC), Word position (WP), Word Frequency (WF), Word Relatedness to Context (WRC), and Word DifSentence (WF). The relation between five features can be expressed through equation (1), where $S(w)$ is the measure for each word. After calculating the measure for each word, the final keyword is calculated utilizing a 3-gram model [15]:

$$S(w) = \frac{\mathrm{WR} * \mathrm{WP}}{\mathrm{WC} + \mathrm{WF}/\mathrm{WRC} + \mathrm{WD}/\mathrm{WR}}. \quad (1)$$

(2) TopicRank. Bougouin et al. proposed TopicRank [16] in 2013, which is a clustering-based model. It divides the document into multiple topics employing the hierarchical agglomerative clustering [17]. Then, utilizing the PageRank [18], it scores each topic and selects each top-ranked candidate keyword from each topic. After that, it selects all the top candidate words as final keywords.

(3) MultipartiteRank. MultipartiteRank is a topic-based keyword extraction model. It encodes topical information of a document in a multipartite graph structure. This technique represents candidate keywords and topics of a document in a single graph, and utilizing the mutually reinforcing relationship of the candidate keywords and topics improves candidate ranking. This method has two steps of selecting candidate words as keywords, (i) representing the whole document in a graph and (ii) assigning relevance score to each word. Between these two steps, position information is captured utilizing edge weights' adjustment. As a result, most of the time, it outperforms different other key-phrase extraction techniques [19].

(4) KPMiner. El-Beltagy and Rafea proposed the KPMiner [20] in 2009. This method also utilizes TF-IDF to calculate words as keywords. This calculation is done in three steps, (i) selecting candidate words from the document utilizing least allowable seen frequency (lasf) factor and CutOff factor, (ii) calculating candidate word's score, and (iii) selecting the candidate word with the highest score utilizing the candidate word position and TF-IDF score as the final keyword.

2.1.2. Supervised Keyword Extraction. While unsupervised algorithms do not need a large amount of labeled training data, supervised algorithms need a large amount of that data and perform poorly except in the training domain. However, for any specific domain, supervised techniques are preferred over unsupervised techniques [15]. In this paper, two supervised techniques are employed, KEA and WINGNUS.

(1) KEA. KEA is a supervised keyword extraction algorithm proposed by Witten et al. in 1999 [21]. KEA classifies a candidate keyword utilizing word frequency and position of the word in the document. After that, it predicts which candidate words are qualified as keywords utilizing the Naive Bayes machine learning algorithm. The machine learning model builds a predictive model initially. Then, keywords are extracted utilizing this predictive model [22].

(2) WINGNUS. This supervised keyword extraction technique is developed focusing on keyword extraction from scientific documents [23]. It utilizes inferred document logical structure [24] in the candidate word identification process to limit the phrase number in the candidate word list. This method utilizes regular expression rules to extract candidate words, and instead of whole document text, it utilizes input text in different levels such as title and headers or abstract and introduction. Like KEA, it also utilizes the Naive Bayes machine learning algorithm to select candidate words.

2.2. Text Similarity Index. Determining how similar two pieces of text are to each other is the simple idea of text similarity index or text similarity calculation. In this study, keywords from different documents extracted by keyword extraction algorithms and expert-provided keywords' similarity are measured. In two ways, this similarity can be measured: one is lexical similarity and another is semantic similarity [25–30]. This paper implemented both the similarity measures utilizing Jaccard, Cosine, and Cosine with word vector similarity indexes and presented the outcome for EDLC-based scientific articles.

2.2.1. Jaccard Similarity. Jaccard similarity index is a lexical similarity index method, which calculates the similarity index at the word level. As lexical similarity is unaware of the word's actual meaning or the entire phrase, Jaccard similarity takes two sets of text and calculates the similarity between all pairs of sets. Jaccard provides a similarity score with a range of 0% to 100%. This algorithm is very sensitive to sample size and may provide unexpected results for a small sample size. Conversely, for larger sample sizes, it is computationally costly [31, 32]. Jaccard similarity index is calculated utilizing equation (2), where A and B are two different sets of text or documents:

$$J(A, B) = \frac{|A \cap B|}{|A| + |B| - |A \cup B|}. \tag{2}$$

2.2.2. Cosine Similarity. The cosine similarity index measures the similarity between two documents utilizing the cosine angle between two multidimensional vectors in a multidimensional space regardless of their size. In this technique, sentences are converted into vectors utilizing the bag of words method and then employing equation (3), where A and B are two documents converted into vectors. This algorithm is computationally expensive for larger data sample [9, 10]:

$$\cos(A, B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \sqrt{\sum_{i=1}^{n} (B_i)^2}}. \tag{3}$$

2.2.3. Word Vector. Word vectors are a type of word embedding, where similar meaningful words are arranged in a similar representation, mostly with vectors. Each word is mapped to a vector in a predefined vector space [33]. It is different from Jaccard similarity in the way that Jaccard measures lexical similarity, but in word vector, it is measured for semantic similarity. Utilizing word vectors, similar meaningful words can be measured rather than the exact word, enabling better scores for similarity measures. In this study, as a word vector model, Wod2vec [11] proposed by Mikolov et el. is utilized. Word2vec is different from the traditional tf-idf measure, where tf-idf sets one number per word, but Word2vec sets one vector per word.

3. Methodology

This study diverges into three major components: (i) data collection, (ii) data processing, and (iii) similarity score calculation. In the data collection component, ground truth data and test data are collected from respective sources. Collected data are cleaned and processed for the similarity calculation component which is done in the data processing component. In the similarity score calculation component, similarity scores for collected data are calculated with different similarity indexes employing different keyword extraction techniques. The conceptual overview of the employed methodology can be found in Figure 1.

3.1. Data Collection. In this study, the electric double layer capacitor (EDLC) domain is considered as the experiment's use case. Hence, from the domain experts, a set of 32 keywords of the EDLC domain has been collected as ground truth keywords, and ten scientific documents are collected from the same domain, which satisfies the keywords and is suggested as the relevant document to the domain. The experiment is based on the quest that, from these ten documents, keywords are extracted through different keyword extraction techniques, and then, extracted keywords are compared for the similarity score with the domain expert-provided keywords. First column from the left of Table 1 contains the domain expert-provided keywords for the EDLC domain. All the scientific documents are collected in portable document format (pdf), and keywords are collected in the plain text.

3.2. Data Processing. In the data processing stage, collected pdf files are initially converted to plain text format. To convert the files, grobid [34] tool is utilized, which primarily converts the pdf files to tei xml format, and then, with a custom tei xml, parser xml contents are converted to a plain

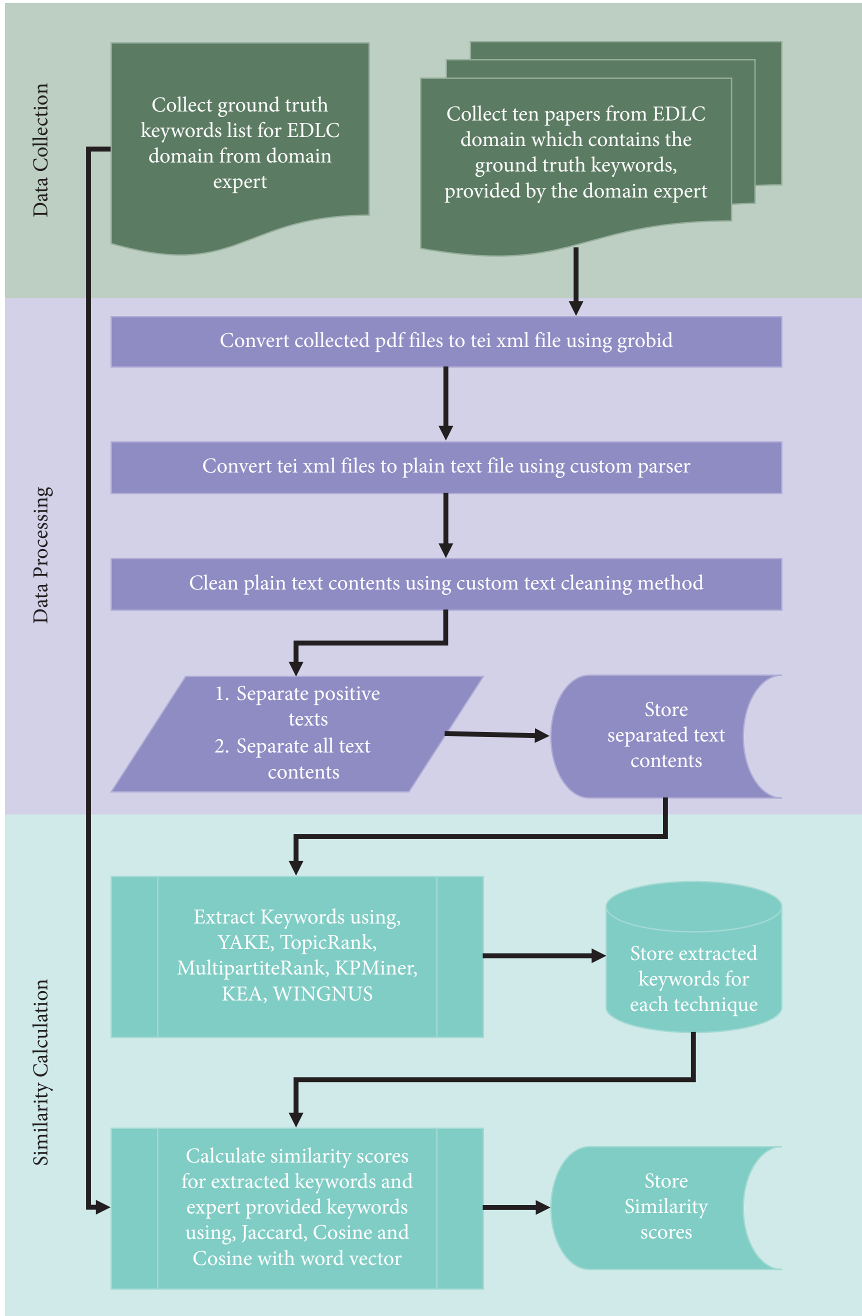


FIGURE 1: Overview of the employed methodology.

text file. The custom xml parser is developed by the authors utilizing the python programming language. After the conversion, text contents are cleaned to remove extra spaces, special characters, extra line breaks, parentheses, references, figures, and tables employing a custom data cleaning method also developed by the authors.

Text cleaning methods are dependent on the dataset and desired output. However, apart from the dataset and output, several steps are commonly performed to clean text data, namely, removing punctuation, filtering out stop words, stemming and lemmatisation, and converting text to upper and lower case. For the dataset used in this study, some of the common cleaning tasks are implemented, and some of them are avoided. In addition to these tasks, some dataset-specific cleanup tasks are also performed. Based on the cleanup activities performed in the dataset, the cleaning process is described as a custom text cleaning process. For example, normalization of nonstandard words (NSW) is not performed in the text cleaning process. NSW are words that are not available in a dictionary, such as numbers, dates, abbreviations, chemical symbols of materials, currency amounts, and acronyms [35]. Most scientific papers contain these NSWs, and they refer to specific processes or operations of any domain which are not available on a dictionary, e.g., “MnO2,” a chemical symbol for a material called manganese dioxide. Stemming and lemmatisation operations on the words are also discarded since most keywords are a combination of several words, e.g., “Helmholtz double layer,” which gives the same result when lemmatised and a meaningless result when stemmed. Table 1 represents the original keywords with the lemmatised and stemmed version of the keywords. From Table 1, it can be observed that the output of the lemmatised keywords is almost similar to the original keywords, and the stemmed version of the keywords produces unintelligible words. In the dataset-specific cleaning process, all tabular data, references, and images are removed from the articles. Then, the text contents are decoded from the UTF8 encoding format. In addition to normalizing these decoded text contents, some special character substitution operations are performed.

Then, from the cleaned text of each document, texts are separated into positive sentences only and all text of the document. For each document, these two types of texts are stored for the similarity calculation component. Positive sentences are identified utilizing negatives and negation-grammar rules [36–38]. There are 2840 sentences in the dataset utilized in this study. Among 2840 sentences, 2240 sentences are positive sentences. Figure 2 represents the overview of the dataset stating the number of total positive and negative sentences. The dataset can be requested through the GitHub repository (https://github.com/ping543f/kwd-extraction-study).

3.3. Similarity Calculation. With two sets of text obtained from the data processing component, all keyword extraction algorithms are employed to extract keywords from each set of each document. Firstly, texts are passed into all the keyword extraction techniques, namely, YAKE, TopicRank, MultipartiteRank, KPMiner, KEA, and WINGNUS. All techniques return the extracted keywords of the provided texts of a document. Then, those keywords and expert-provided keywords are passed to the similarity index calculator to calculate the similarity score between them. Three similarity indexes are utilized to calculate the similarity score, namely, Jaccard, Cosine, and Cosine with word vector similarity index. This whole process is executed for all the documents with positive and all texts of each document. After processing each document, scores are stored with appropriate labels to analyze the result. The similarity calculation component for the scenario described above can be expressed through Algorithm 1.

3.4. Experimental Setup. All experiment-related codes are developed utilizing Python programming language version 3.7.3 [39] for this study. Jaccard and cosine similarity algorithms are developed following the equation described in [8, 40]. Cosine similarity with word vector algorithm is implemented utilizing Spacy Python library [41]. All keyword extraction algorithms are implemented utilizing pke [42] Python package. The experiment is done in a MacBook with macOS Big Sur operating system version 11.5 with a 1.2 GHz dual-core Intel Core m5 processor and 8 gigabytes of RAM.

4. Results and Discussion

To begin with the result analysis, Tables 2 and 3 are generated from the experiment. Both tables contain the similarity scores of ten standard documents generated by different keyword extraction techniques and similarity index algorithms. Table 2 contains the results obtained from the unsupervised keyword extraction techniques, and Table 3 contains the results generated by the supervised keyword extraction techniques. For unsupervised techniques, the MultipartiteRank algorithm performs better in all three similarity indexes than other implemented keyword extraction techniques. Furthermore, it gives the best result of 92% similarity score for positive sentences and 91% for all sentences of the documents while employed with the cosine with word vector similarity index. The lowest performing similarity index algorithm is the Jaccard similarity index for the same keyword extraction technique with a score of 14% similarity score for both positive and all sentences of the documents. It is also observed from the experimental result that cosine with word vector similarity index is consistently performing better than Jaccard and cosine similarity index for all the unsupervised keyword extraction techniques. This analysis can easily be understood from Figure 3(a). This figure presents the distribution of all the similarity scores of all the unsupervised techniques employed in this study for Jaccard, cosine, and cosine with word vector similarity indexes.

On the contrary, for the supervised techniques, the KEA keyword extraction algorithm performs the best with 91% of similarity score while calculating with the cosine with word vector similarity index for both positive and all sentences of the documents. However, the WINGNUS supervised keyword extraction technique provides better similarity scores for cosine and Jaccard similarity indexes only for positive sentences, which are 22% and 12% similarity scores. Nevertheless, KEA is performing better for all sentences while measured with Jaccard and cosine similarity indexes. However, KEA holds the best similarity score utilizing the cosine with word vector similarity index, which is around 70% more than those measured with Jaccard and cosine similarity index. This analysis can be more clear with a visual representation. Figure 3(b) represents the distribution of all the similarity scores for all the supervised keyword extraction techniques with all three similarity indexes.

Among supervised and unsupervised keyword extraction techniques, the unsupervised technique, namely, MultipartiteRank, exhibits better performance in achieving a higher similarity score for positive sentences while measured with cosine with word vector similarity index. Furthermore, for all sentences, unsupervised technique, MultipartiteRank, and supervised technique, KEA produces the same score of 91% in cosine with word vector similarity index. Similarity score comparisons for both supervised and unsupervised methods are projected in Figure 4.

Table 1: Domain expert-curated keywords for EDLC domain with lemmatised and stemmed version. From left, keywords' column contains the original keywords provided by the domain experts. Lemmatised keyword and stemmed keyword columns contain lemmatised and stemmed version of the original keywords.

Keyword	Lemmatised keyword	Stemmed keyword
Supercapacitors	Supercapacitors	Supercapacitors
scs	sc	sc
Electrochemical capacitors	Electrochemical capacitors	Electrochemical capacitor
Energy storage device	Energy storage device	Energy storage device
Electric double-layer capacitor	Electric double-layer capacitor	Electric double-layer capacitor
edlc	edlc	edlc
Pseudocapacitance	Pseudocapacitance	Pseudocapacitance
Electrostatic adsorption	Electrostatic adsorption	Electrostatic adsorption
Electrosorption	Electrosorption	Electrosorption
Faradaic redox reactions	Faradaic redox reactions	Faradaic redox react
Stern layer	Stern layer	Stern lay
Helmholtz double layer	Helmholtz double layer	Helmholtz double lay
Double-layer formation	Double-layer formation	Double-layer formation
Activated carbon	Activated carbon	Activated carbon
Porous carbon	Porous carbon	Porous carbon
Carbon nanotubes	Carbon nanotubes	Carbon nanotubes
Graphene	Graphene	Graphene
Graphite oxide	Graphite oxide	Graphite oxide
go	go	go
Reduced graphite oxide	Reduced graphite oxide	Reduced graphite oxide
rgo	rgo	rgo
Surface charge accumulation	Surface charge accumulation	Surface charge accumulation
High power applications	High power applications	High power applications
Charge separation at electrode interface	Charge separation at electrode interface	Charge separation at electrode interface
Charge separation at electrolyte interface	Charge separation at electrolyte interface	Charge separation at electrolyte interface
Nonfaradaic process	Nonfaradaic process	Nonfaradaic process
Specific surface area	Specific surface area	Specific surface area
Pore size distribution	Pore size distribution	Pore size distribution
Electrochemical interface	Electrochemical interface	Electrochemical interface
edlc characteristics	edlc characteristics	edlc characteristics
Diffuse double layer	Diffuse double layer	Diffuse double lay
Polarizable capacitor electrode	Polarizable capacitor electrode	Polarizable capacitor electrode

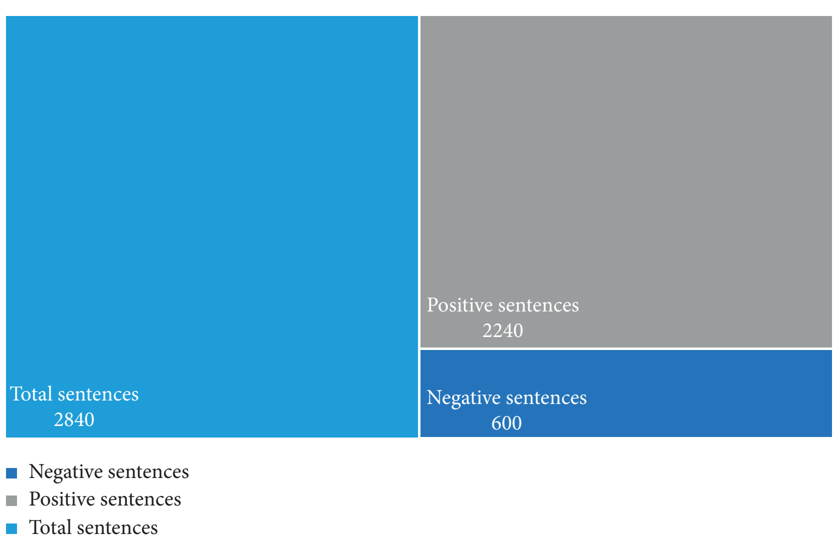


Figure 2: Positive and negative sentence distribution of the dataset utilized in this study.

Since there are two sets of textual data, data with positive sentences and data with all sentences, they have implications for the experimental results seen in Tables 2 and 3. The initial hypothesis of having two separate text datasets from the same articles is to observe how positive and negative sentences affect the similarity score of the extracted keywords

```
Input: Whole text String A_string
Input: Positive sentence String P_string
Input: Domain expert-curated keywords list's string KW_string
Output: String containing filename, algorithm and score
Def get_score(Sim_algo, KPalgo_name, text_content, KW_string):
  score = 0
  algo_list = ["yake", "topicrank", "multipartiterank",
  "kpminer", "kea", "wingnus"]
  if KPalgo_name in algo_list then
    algo_name = KPalgo_name keyWords = Extract Keywords using algo_name algorithm from text_content
    SimScore = Calculate similarity of keyWords with KW_string using Sim_algo
    score = SimScore
    return score
  end
  else
    return error_msg
  end
Def main(Kw_args):
  sim_algo = [jaccard, cosine, coswv]
  algorithm_list = ["yake", "topicrank", "multipartiterank", "kpminer", "kea", "wingnus"]
  for algo in sim_algo do
    for algorithm in algorithm_list do
      score_a = get_score (algo, algorithm, A_string, KW_string)
      score_p = get_score (algo, algorithm, P_string, KW_string)
      r_string = algo + algorithm + score_a + score_p
    end
    return r_string
  end
```

Algorithm 1: Similarity score calculation.

Table 2: Similarity scores calculated for different unsupervised keyword extraction techniques.

	YAKE		
	Jaccard	**Cosine**	**Cosine with word vector**
Positive sentence	0.10	0.20	0.83
All sentence	0.10	0.21	0.87
	TopicRank		
	Jaccard	**Cosine**	**Cosine with word vector**
Positive sentence	0.13	0.23	0.91
All sentence	0.11	0.19	0.90
	MultipartiteRank		
	Jaccard	**Cosine**	**Cosine with word vector**
Positive sentence	0.14	0.25	0.92
All sentence	0.14	0.25	0.91
	KPMiner		
	Jaccard	**Cosine**	**Cosine with word vector**
Positive sentence	0.10	0.19	0.88
All sentence	0.11	0.21	0.89

Table 3: Similarity scores calculated for different supervised keyword extraction techniques.

	KEA		
	Jaccard	**Cosine**	**Cosine with word vector**
Positive sentence	0.11	0.20	0.91
All sentence	0.11	0.21	0.91
	Wingnus		
	Jaccard	**Cosine**	**Cosine with word vector**
Positive sentence	0.12	0.22	0.87
All sentence	0.11	0.20	0.88

with the keywords provided by the experts for the specific domain, and based on this impact, we recommend the relevant text data to be used. From the experimental results, the positive sentences have a minimal impact on the similarity scores for all three similarity indices compared to the scores for all sentences. This is because the negative sentences contain very few to no keywords that could match the keywords given by the experts. Therefore, there is no or minimal effect of the similarity indices between the positive sentences and the dataset with all sentences, as shown in the experimental result. The similarity values between the positive sentences and all sentences vary from 1% to 4%. For example, in the MultipartiteRank algorithm, the Jaccard and cosine similarity values are the same for both texts, 14% and 25%, respectively. However, for the cosine with word vector similarity index, the text of the positive sentence achieves 92% similarity, and the text of all sentences achieves 91% similarity, which is a minimal difference of 1%. On the other hand, in the algorithm KEA, the similarity value of cosine with word vector is the same for both text data, i.e., 91% of

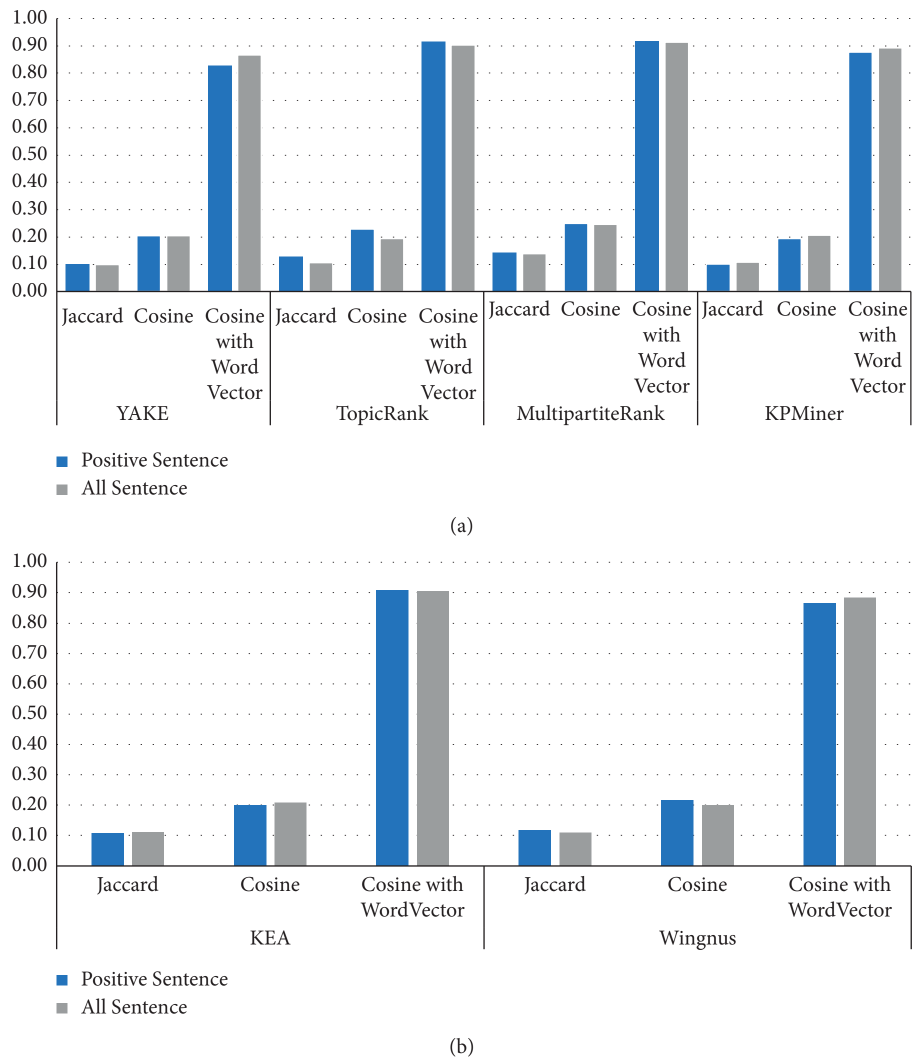


Figure 3: Distribution of similarity scores of supervised and unsupervised keyword extraction techniques employed in positive and all sentences for Jaccard, cosine, and cosine with Word vector similarity indexes. (a) Similarity score distribution of positive and all sentences for unsupervised YAKE, TopicRank, MultipartiteRank and KPMiner keyword extraction algorithms for all the similarity indexes. (b) Similarity score distribution of positive and all sentences for supervised KEA and Wingnus keyword extraction algorithms for all the similarity indexes.

the similarity value. The maximum difference of 4% in similarity score is observed for the YAKE algorithm in similarity index cosine with Word vector. Hence, it can be said that positive sentences and all sentences have a similar effect on the similarity index with very little difference from 1% to 4%.

Although the positive sentences have a negligible effect on the similarity computation, they have a more significant impact on the running time of the similarity computation process. From the experiment results, the unsupervised algorithms MultipartiteRank and the supervised algorithms KEA perform better than the other algorithms used in terms of similarity index. Therefore, a runtime comparison is performed for both algorithms to study the runtime for both positive and all text sets for computing all similarity indices. Table 4 presents the runtime comparison result for the two better-performing keyword extraction techniques MultipartiteRank and KEA for Jaccard, cosine, and cosine similarity with Word vector indices. The runtimes reported in Table 4 are the average of 5 runtimes of the experiment, which includes only the similarity computation. From the runtime table, it can be seen that positive texts have a great impact on the duration of the similarity calculation. When computing the similarity of the texts with the keywords given by the experts, the positive sentences take significantly less time than computing the similarity of all sentences. For example, in the unsupervised MultipartiteRank algorithm, the computation of all sentences takes 232.4, 225.1, and 230.2 seconds for the Jaccard, cosine, and cosine with Word vector similarity indices, respectively. On the contrary, the computation of positive sentences takes only 143.6, 140.86, and 142.7 seconds for Jaccard, cosine, and cosine with Word vector similarity indices, respectively, which is 88.8, 84.24, and 87.5 seconds less for the aforementioned similarity

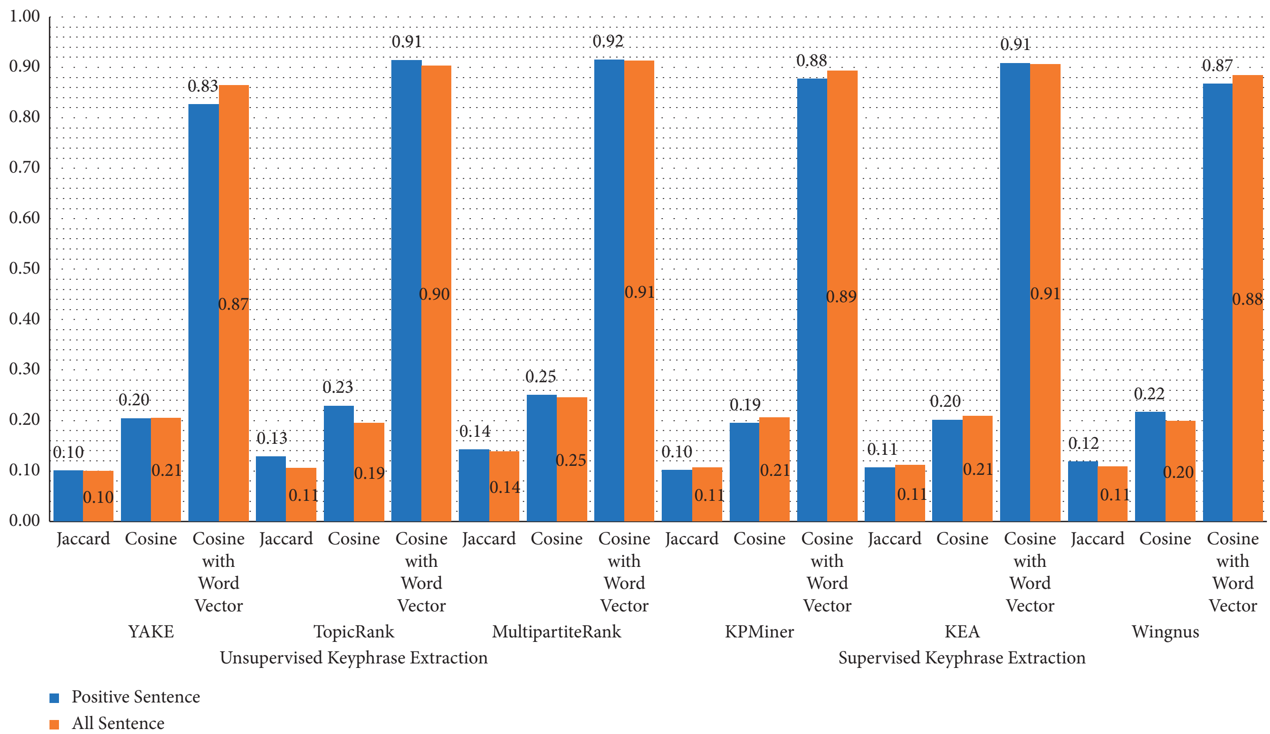


Figure 4: Similarity scores of different supervised and unsupervised keyword extraction techniques for Jaccard, cosine, and cosine with Word vector similarity indexes.

Table 4: Runtime comparison in seconds (s) of positive and all sentences' texts for MultipartiteRank and KEA keyword extraction algorithms in terms of Jaccard, cosine, and cosine with Word vector similarity indexes.

	Jaccard	Cosine	Cosine with word vector
MultipartiteRank all sentences	232.4 s	225.1 s	230.2 s
MultipartiteRank positive sentences	143.6 s	140.86 s	142.7 s
KEA all sentences	97.1 s	96.28 s	96.65 s
KEA positive sentences	93.5 s	92 s	91.72 s

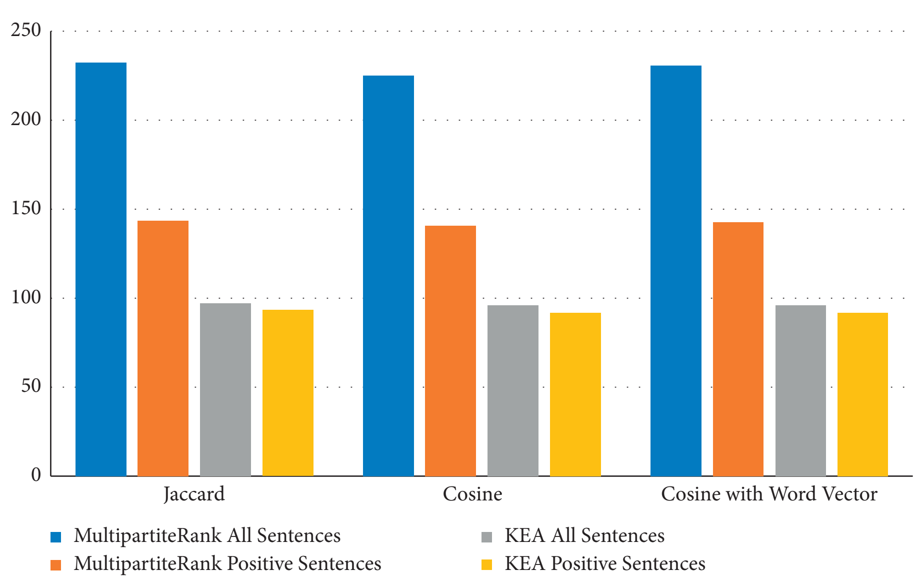
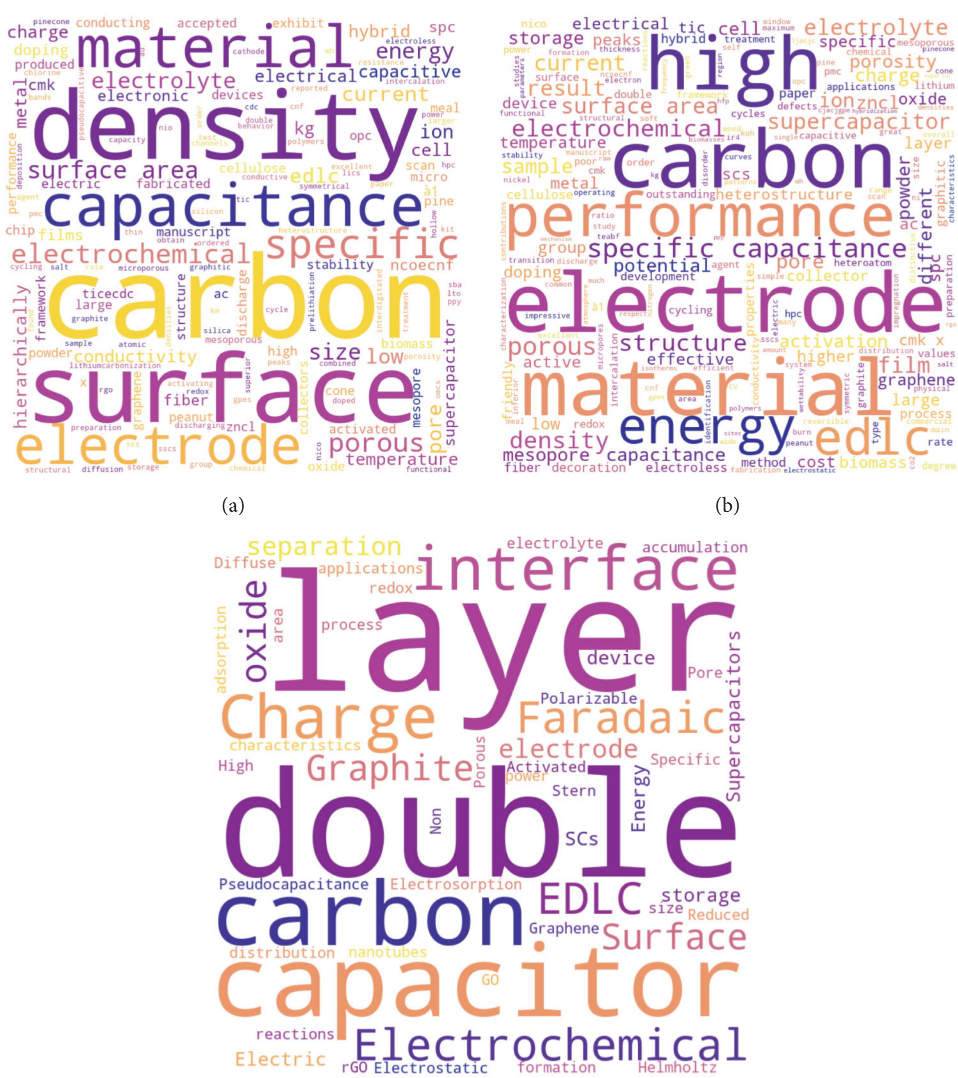


Figure 5: Comparative scores of similarity calculation run times for positive and all sentences employing MultipartiteRank and KEA keyword extraction algorithms.

Table 5: Sample keywords extracted by MultipartiteRank, KEA keyword extraction techniques, and domain expert-curated keywords.

Domain expert-curated keywords	MultipartiteRank extracted keywords	KEA extracted keywords
Supercapacitors, scs,electrochemical capacitors, energy storage device, electric double-layer capacitor, edlc, pseudocapacitance, electrostatic adsorption, electrosorption, faradaic redox reactions, stern layer, Helmholtz double layer, double-layer formation, activated carbon, porous carbon, carbon nanotubes, graphene, graphite oxide, go, reduced graphite oxide, rgo, surface charge accumulation, high-power applications, charge separation at electrode interface, charge separation at electrolyte interface, nonfaradaic process, specific surface area, pore size distribution, electrochemical interface, edlc characteristics, diffuse double layer, and polarizable capacitor electrode	Layer, power, scs, charge, formation, high energy, chemical, graphene, surface area, porous carbon, ions, electrolyte, rgo, graphite, energy storage, carbon, electrochemical, surface, pore size distribution, electrode, edlc, supercapacitor, adsorption, supercapacitors, device, and capacitance	scs, charge, pore, energy, redox, size, chemical, graphene, ion, surface area, porous carbon, ions, electrolyte, pore size, rgo, graphite, energy storage, carbon, electrochemical, surface, electrode, edlc, specific surface, supercapacitor, porous, specific surface area, oxide, supercapacitors, electric, double, tic, and capacitance



(a)

(b)

(c)

Figure 6: Word cloud representation of the keywords extracted by the top-performing keyword extraction techniques achieved with cosine with Word vector similarity index. (a) Word cloud of the keywords extracted by supervised. (b) Word cloud of the keywords extracted by unsupervised technique MultipartiteRank. (c) Word cloud of the keywords provided by EDLC domain.

indices. A similar pattern is also observed for the supervised KEA algorithm, i.e., computing the similarity of positive sentences takes less time than computing all sentences. Figure 5 shows the comparison results in a more understandable form.

Table 5 provides the set of keywords extracted by the top-performing keyword extraction techniques employing the cosine with Word vector similarity index and expert-provided keywords. This table also provides a visual comparison of the similarity between all the keywords. Word cloud representation is also provided in Figure 6. Word cloud is utilized to represent the words emphasized according to their frequency, rank, or similarity. This word cloud is generated based on the frequency scores of keywords among all the documents. From the word clouds of top-performing two methods, it is also visible that there are similar keywords of the same scores among all machine-generated and expert-provided keywords.

The study of the experimental results suggests that, for extracting keywords and checking the similarity of the extracted keywords from scientific documents, especially for the EDLC-related documents, the unsupervised keyword extraction technique MultipartiteRank algorithm can be considered in addition to the expert-curated keywords. Although this algorithm requires slightly more computation time than the supervised keyword extraction technique KEA, it gives better results than KEA. If computation time is considered or required over better similarity score, then it is recommended to employ the supervised keyword extraction technique KEA for 1% of similarity score drop over MultipartiteRank algorithm. When choosing between the positive and the whole article text content, it is recommended to choose the positive text as it has a very small impact on the similarity score but a larger impact on the computation time. Positive texts have no or very little impact on the similarity scores, but require less computation time than all the texts of the scientific articles.

5. Conclusion

The aim of this study is to find out which keyword extraction technique provides more similar keywords to the expert-provided keywords, which text types have more similarity, which similarity index provides more similarity scores, and whether the use of machine-generated keywords is feasible with respect to the expert-provided keywords. The experiment shows that the unsupervised keyword extraction technique MultipartiteRank provides 92% similarity with the expert-provided keywords in cosine with the Word vector similarity index for positive sentences of the documents from EDLC domain. This study can be further extended with keywords for other domains with a larger dataset in other environments, including author-supplied keywords.

Data Availability

The dataset used in this study is available from the corresponding author upon request and the request repository is mentioned in Section 3.2.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] S. Rose, D. Engel, N. Cramer, and W. Cowley, "Automatic keyword extraction from individual documents," *Text Mining*, vol. 1, pp. 1–20, 2010.

[2] K. S. Hasan and V. Ng, "Automatic keyphrase extraction: a survey of the state of the art," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pp. 1262–1273, Baltimore, MA, USA, June 2014.

[3] M. Saef Ullah Miah, M. Sadid Tahsin, S. Azad et al., "A geofencing-based recent trends identification from twitter data," *IOP Conference Series: Materials Science and Engineering*, vol. 769, no. 1, Article ID 012008, 2020.

[4] T. B. Sarwar and N. M. Noor, "An experimental comparison of unsupervised keyphrase extraction techniques for extracting significant information from scientific research articles," in *Proceedings of the 2021 International Conference on Software Engineering & Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM)*, pp. 130–135, IEEE, Pekan, Malaysia, August 2021.

[5] M. S. U. Miah, M. S. Tahsin, S. Azad et al., "A geofencing-based recent trends identification from twitter data," in *Proceedings of the IOP Conference Series: Materials Science and Engineering*, IOP Publishing, Chennai, India, September 2020.

[6] M. S. U. Miah, J. Sulaiman, S. Azad, K. Z. Zamli, and R. Jose, "Comparison of document similarity algorithms in extracting document keywords from an academic paper," in *Proceedings of the 2021 International Conference on Software Engineering & Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM)*, pp. 631–636, IEEE, Pekan, Malaysia, August 2021.

[7] S. Beliga, *Keyword Extraction: A Review of Methods and Approaches*, University of Rijeka, Department of Informatics, Rijeka, Croatia, 2014.

[8] P. Jaccard, "The distribution of the flora in the alpine zone.1," *New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[9] "Cosine Similarity-understanding the math and how it works? (with python)," https://www.machinelearningplus.com/nlp/cosine-similarity/.

[10] 9.5.2. The Cosine Similarity Algorithm-9.5. Similarity Algorithms, https://neo4j.com/docs/graph-algorithms/current/labs-algorithms/cosine/.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, https://arxiv.org/abs/1301.3781.

[12] N. Firoozeh, A. Nazarenko, F. Alizon, and B. Daille, "Keyword extraction: issues and methods," *Natural Language Engineering*, vol. 26, no. 3, pp. 259–291, 2020.

[13] K. Bennani-Smires, C. Musat, A. Hossmann, M. Baeriswyl, and M. Jaggi, "Simple unsupervised keyphrase extraction using sentence embeddings," in *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pp. 221–229, Association for Computational Linguistics, Brussels, Belgium, October 2018.

[14] R. Campos, V. Mangaravite, A. Pasquali, A. Jorge, C. Nunes, and A. Jatowt, "Yake! Keyword extraction from single

documents using multiple local features," *Information Sciences*, vol. 509, pp. 257–289, 2020.

[15] C. Sun, L. Hu, S. Li, T. Li, H. Li, and L. Chi, "A review of unsupervised keyphrase extraction methods using within-collection resources," *Symmetry*, vol. 12, no. 11, pp. 1–20, 2020.

[16] A. Bougouin, F. Boudin, and B. Daille, "Topicrank: graph-based topic ranking for keyphrase extraction," in *Proceedings of the International joint conference on natural language processing (IJCNLP)*, pp. 543–551, Nagoya, Japan, October 2013.

[17] O. Medelyan, E. Frank, and I. H. Witten, "Human-competitive tagging using automatic keyphrase extraction," in *Proceedings of the 2009 conference on empirical methods in natural language processing*, pp. 1318–1327, Singapore, August 2009.

[18] L. Page, "Method for node ranking in a linked database," Google Patents, 1997.

[19] F. Boudin, "Unsupervised keyphrase extraction with multipartite graphs," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 667–672, Association for Computational Linguistics, New Orleans, LA, USA, June 2018.

[20] S. R. El-Beltagy and A. Rafea, "Kp-miner: a keyphrase extraction system for English and Arabic documents," *Information Systems*, vol. 34, no. 1, pp. 132–144, 2009.

[21] I. Witten, G. Paynter, E. Frank, C. Gutwin, and C. nevillmanning, "kea: practical automatic keyphrase extraction," in *Proceedings of the Fourth ACM Conference on Digital, Libraries*, pp. 254-255, ACM, Berkeley, CA, USA, August 1999.

[22] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning, "Kea," in *Design and Usability of Digital Libraries: Case Studies in the Asia Pacific*, pp. 129–152, IGI global, Hershey, PA, USA, 2005.

[23] T. D. Nguyen and M.-T. Luong, "Wingnus: keyphrase extraction utilizing document logical structure," in *Proceedings of the 5th international workshop on semantic evaluation*, pp. 166–169, ACM, Los Angeles CA, USA, July 2010.

[24] S. Mao, A. Rosenfeld, and T. Kanungo, "Document structure analysis algorithms: a literature survey," in *Document Recognition and Retrieval X*, vol. 5010, pp. 197–207, International Society for Optics and Photonics, Bellingham, WA, USA, 2003.

[25] G. Maheshwari, P. Trivedi, H. Sahijwani, K. Jha, S. Dasgupta, and J. Lehmann, "Simdoc: topic sequence alignment based document similarity framework," in *Proceedings of the Knowledge Capture Conference*, pp. 1–8, ACM, Austin, TX, USA, December 2017.

[26] C. Yang, B. He, and Y. Ran, "Utilizing embeddings for ad-hoc retrieval by document-to-document similarity," 2017, https://arxiv.org/abs/1708.03181.

[27] S. Aryal, K. M. Ting, T. Washio, and G. Haffari, "A new simple and effective measure for bag-of-word inter-document similarity measurement," 2019, https://arxiv.org/abs/1902.03402.

[28] P. Sitikhu, K. Pahi, P. Thapa, and S. Shakya, "A comparison of semantic similarity methods for maximum human interpretability," in *Proceedings of the International Conference on Artificial Intelligence for Transforming Business and Society, AITB 2019*, Kathmandu, Nepal, November 2019.

[29] V. Thada and V. Jaglan, "Comparison of jaccard, dice, cosine similarity coefficient to find best fitness value for web retrieved documents using genetic algorithm," *International Journal of Innovations in Engineering and Technology*, vol. 2, no. 4, pp. 202–205, 2013.

[30] R. Steinberger, B. Pouliquen, and J. Hagman, "Cross-lingual document similarity calculation using the multilingual thesaurus EUROVOC," *Computational Linguistics and Intelligent Text Processing*, vol. 2276, pp. 415–424, 2002.

[31] I. Neo4j, "9.5.1.The Jaccard Similarity Algorithm-9.5.Similarity Algorithms, https://neo4j.com/docs/graph-algorithms/current/labs-algorithms/jaccard/.

[32] S. Glen, ""Jaccard index/similarity coefficient" from StatisticsHowTo.com: elementary statistics for the rest of us!“," https://www.statisticshowto.com/jaccard-index/.

[33] J. Brownlee, *What Are Word Embeddings For Text?*, https://machinelearningmastery.com/what-are-word-embeddings/.

[34] Grobid, pp. 2008–2021, https://github.com/kermitt2/grobid.

[35] R. Sproat, A. W. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards, "Normalization of non-standard words," *Computer Speech & Language*, vol. 15, no. 3, pp. 287–333, 2001.

[36] Grammarly, "Negatives and negation–grammar rules grammarly," 2021, https://www.grammarly.com/blog/negatives/.

[37] J. Col, "Negative vocabulary word list-enchanted learning," 1998, https://www.enchantedlearning.com/wordlist/negativewords.shtml.

[38] Y. HaCohen-Kerner and H. Badash, "Positive and negative sentiment words in a blog corpus written in Hebrew," *Procedia Computer Science*, vol. 96, pp. 733–743, 2016.

[39] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*, CreateSpace, Scotts Valley, CA, USA, 2009.

[40] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.

[41] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd, *SpaCy: Industrial-Strength Natural Language Processing in Python*, Zenodo, Honolulu, HI, USA, 2020.

[42] F. Boudin, "pke: an open source python-based keyphrase extraction toolkit," in *Proceedings of the COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pp. 69–73, Osaka, Japan, December 2016.